Original ridge Slides

Ridge Regression:

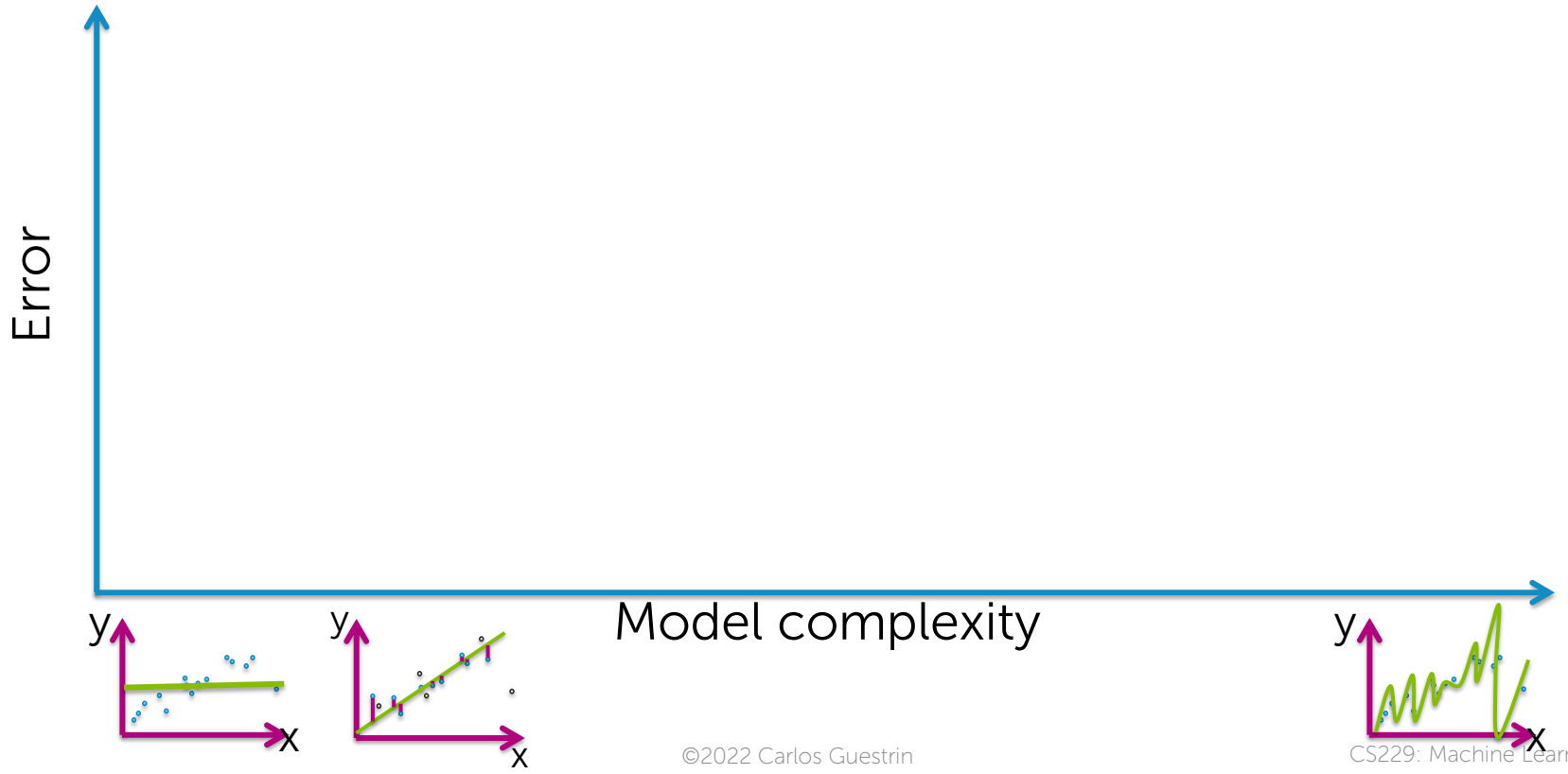
Regulating overfitting when using many features

CS229: Machine Learning

Carlos Guestrin
Stanford University

Slides include content developed by and co-developed with Emily Fox

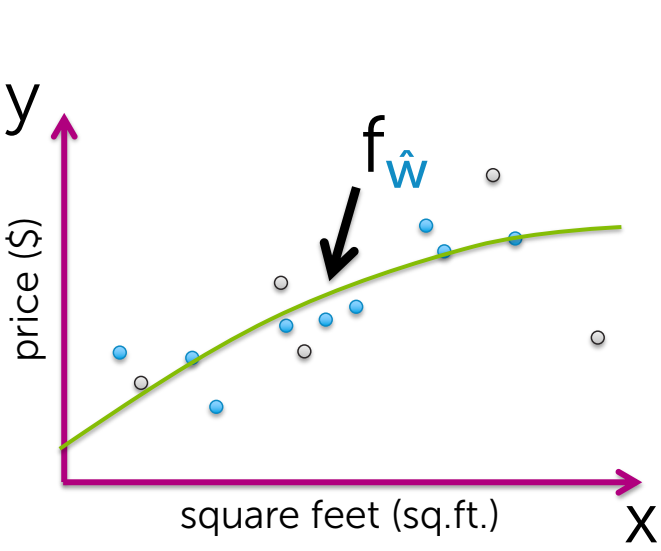
Training, true vs. model complexity

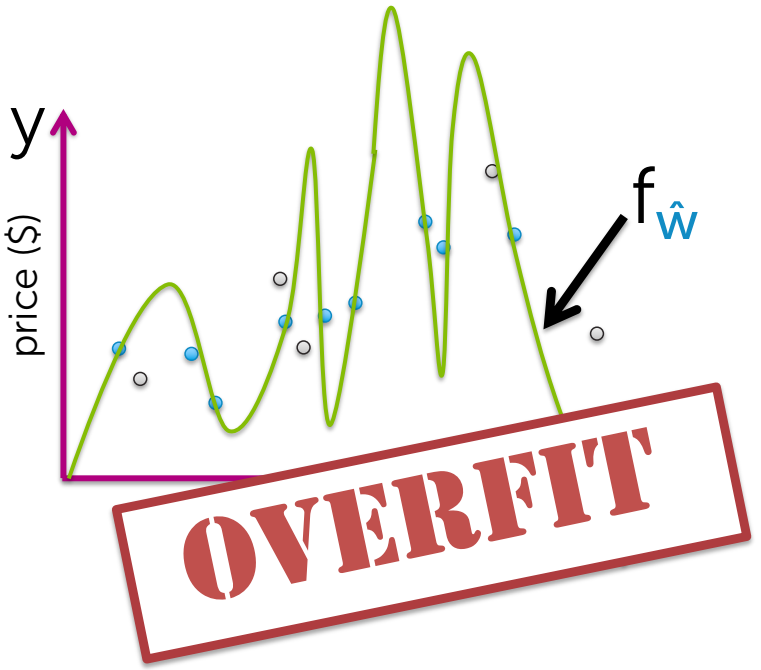


Overfitting of polynomial regression

Flexibility of high-order polynomials

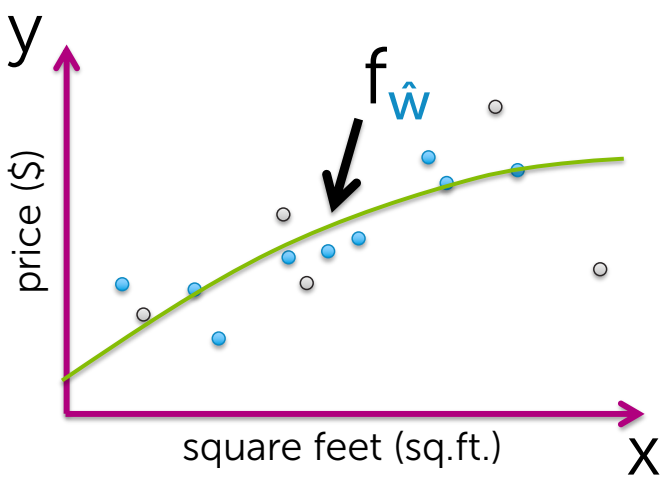
$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p + \varepsilon_i$$

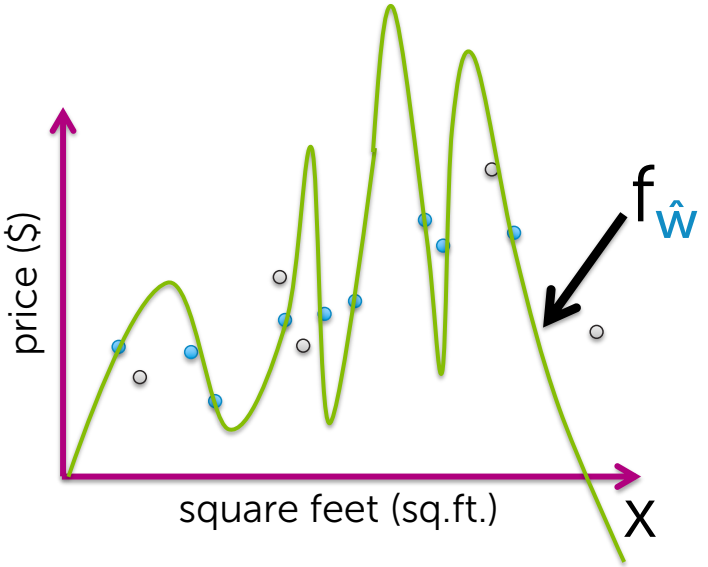




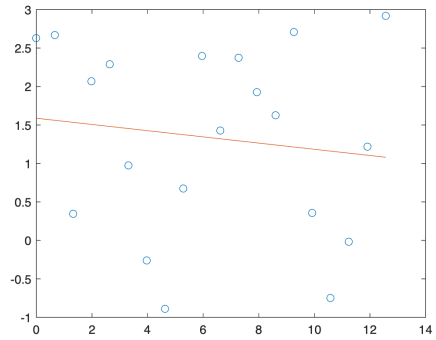
Symptom of overfitting

Often, overfitting associated with very large estimated parameters $\hat{w}$

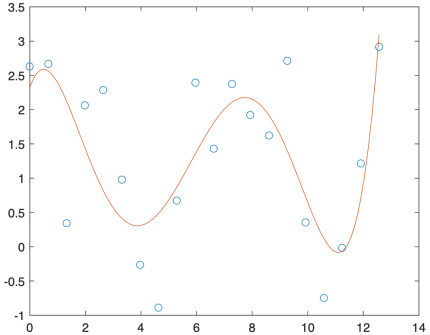




Polynomial fit example

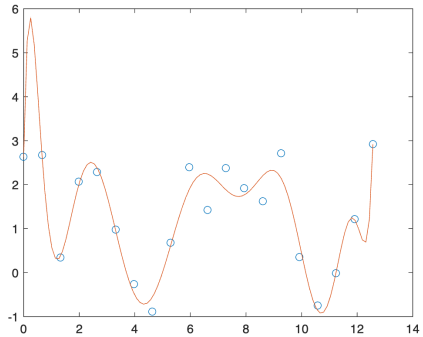


w0 = −0.0403609 w1 = 1.58766



w0 = 0.00142109 w1 = −0.0412048 w2 = 0.402433

w3 = −1.45804 w4 = 1.16305 w5 = 2.32569



w0 = −3.33355e−09 w1 = 3.24407e−07 w2 = −1.3957e−05 w3 = 0.000351859 w4 = −0.00580734

w5 = 0.0664276 w6 = −0.543967 w7 = 3.24647 w8 = −14.1922 w9 = 44.8987 w10 = −98.886

w11 = 139.912 w12 = −109.084 w13 = 32.5699 w14 = 2.62986

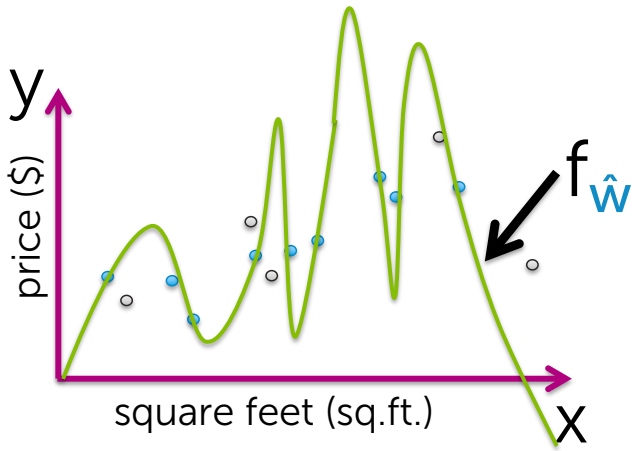
How does # of observations influence overfitting?

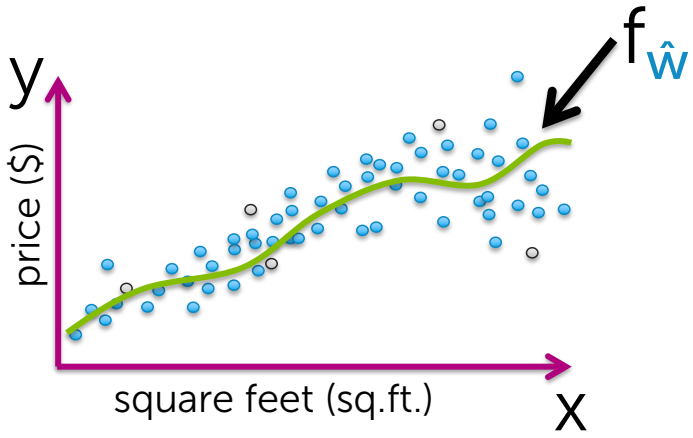
Few observations (N small)

→ rapidly overfit as model complexity increases

Many observations (N very large)

→ harder to overfit





Overfitting of linear regression models more generically

Overfitting with many features

Not unique to polynomial regression,
but also if **lots of inputs** (d large)

- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- ...

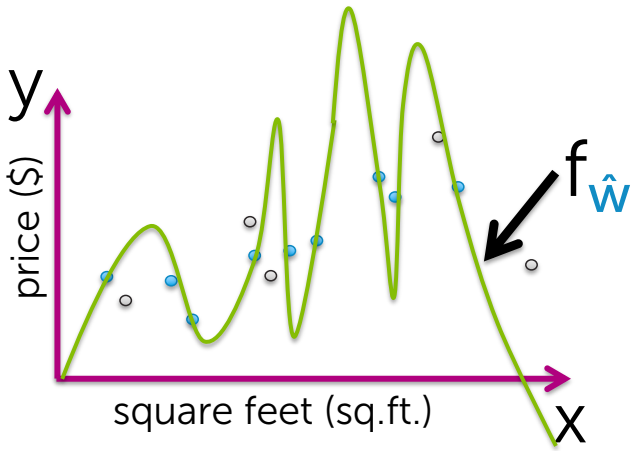
Or, generically,
lots of features (D large)

$$y = \sum_{j=0}^{D} w_j h_j(x) + \varepsilon$$

How does # of inputs influence overfitting?

1 input (e.g., sq.ft.):

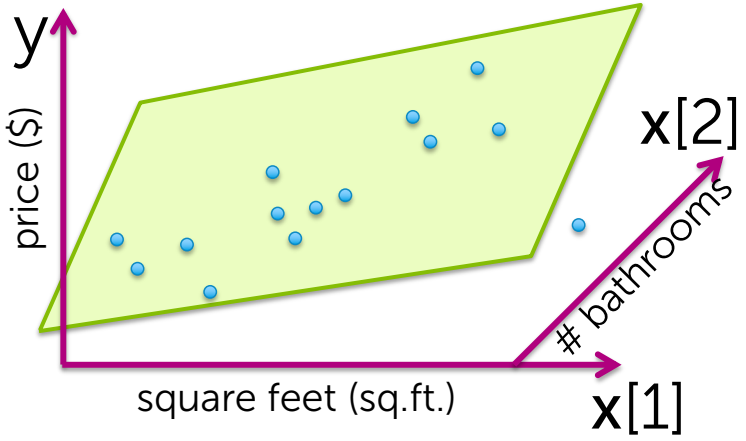
Data must include representative examples of all possible (sq.ft., $) pairs to avoid overfitting



How does # of inputs influence overfitting?

d inputs (e.g., sq.ft., #bath, #bed, lot size, year,…):

Data must include examples of all possible (sq.ft., #bath, #bed, lot size, year,…., $) combos to avoid overfitting



Regularization:
Adding term to cost-of-fit to prefer small coefficients

Desired total cost format

Want to balance:

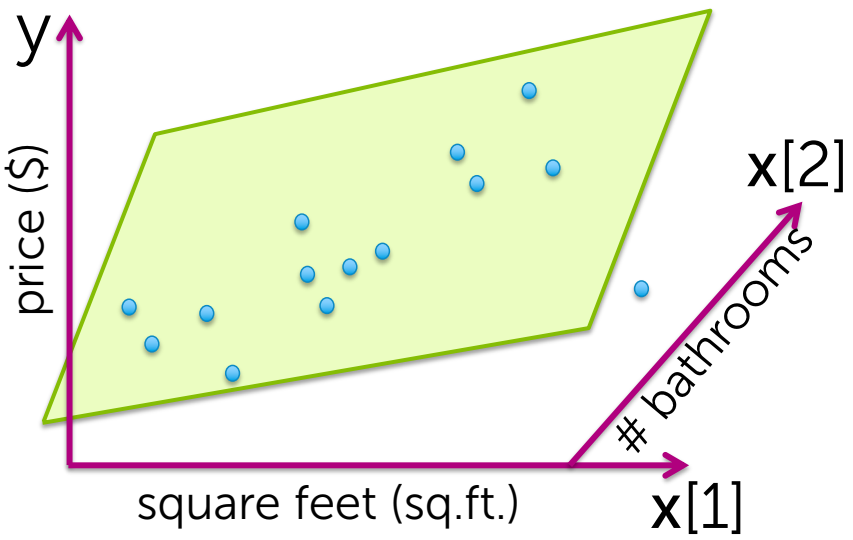
i. How well function fits data

ii. Magnitude of coefficients

Total cost =

measure of fit + measure of magnitude of coefficients

Measure of fit to training data



$$\text{RSS}(\mathbf{w}) \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

Measure of magnitude of regression coefficient

What summary # is indicative of size of regression coefficients?

- Sum?
- Sum of absolute value?
- **Sum of squares (L_2 norm)**

Consider specific total cost

Total cost =

measure of fit + measure of magnitude of coefficients

Ridge Regression (aka L_2 regularization)

What if $\hat{\mathbf{w}}$ selected to minimize

$$\text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|_2^2$$

If λ=0:

If λ=∞:

If λ in between:

Bias-variance tradeoff

Large λ:

bias, variance

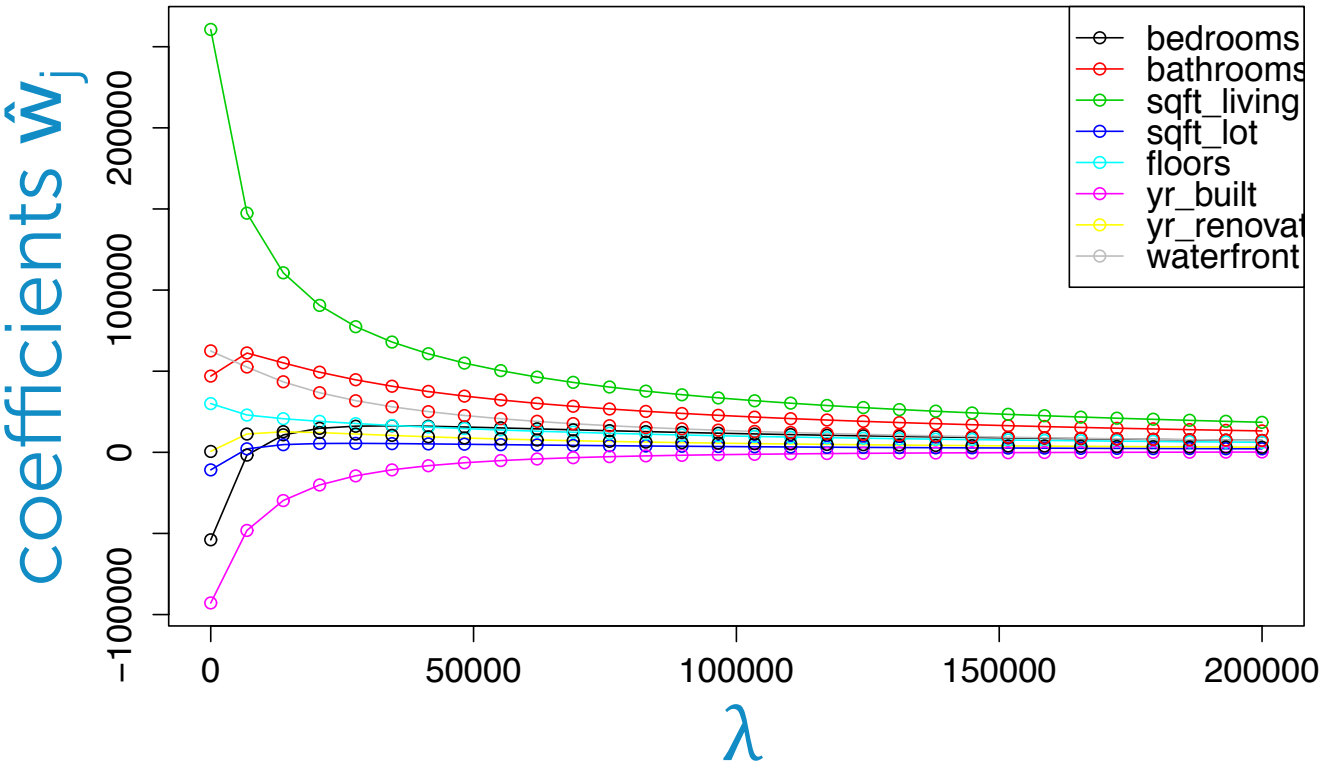
(e.g., $\hat{w} = 0$ for $\lambda = \infty$)

Small λ:

bias, variance

(e.g., standard least squares (RSS) fit of high-order polynomial for $\lambda = 0$)

Coefficient path



How to choose λ

The regression/ML workflow

1. **Model selection**
   Need to choose tuning parameters λ controlling model complexity

2. **Model assessment**
   Having selected a model, assess generalization error

Hypothetical implementation 1

1. **Model selection**

For each considered λ :

i. Estimate parameters $\hat{w}_\lambda$ on **training data**

ii. Assess performance of $\hat{w}_\lambda$ on **training data**

iii. Choose λ^* to be λ with lowest train error

2. **Model assessment**

Compute test error of $\hat{w}_{\lambda^*}$ (fitted model for selected λ^*) to approx. true error

Hypothetical implementation 1

Issue: Both λ and $\hat{w}$ selected on training data then $\lambda^* = 0$

- λ^* was selected to minimize training error (i.e., λ^* was fit on training data)
- Most complex model will have lowest training error

Hypothetical implementation 2

1. **Model selection**

For each considered λ :

i. Estimate parameters $\hat{w}_\lambda$ on **training data**

ii. Assess performance of $\hat{w}_\lambda$ on **test data**

iii. Choose λ^* to be λ with lowest test error

2. **Model assessment**

Compute test error of $\hat{w}_{\lambda^*}$ (fitted model for selected λ^*) to approx. true error

Hypothetical implementation 2

Issue: Just like fitting $\hat{w}$ and assessing its performance both on training data

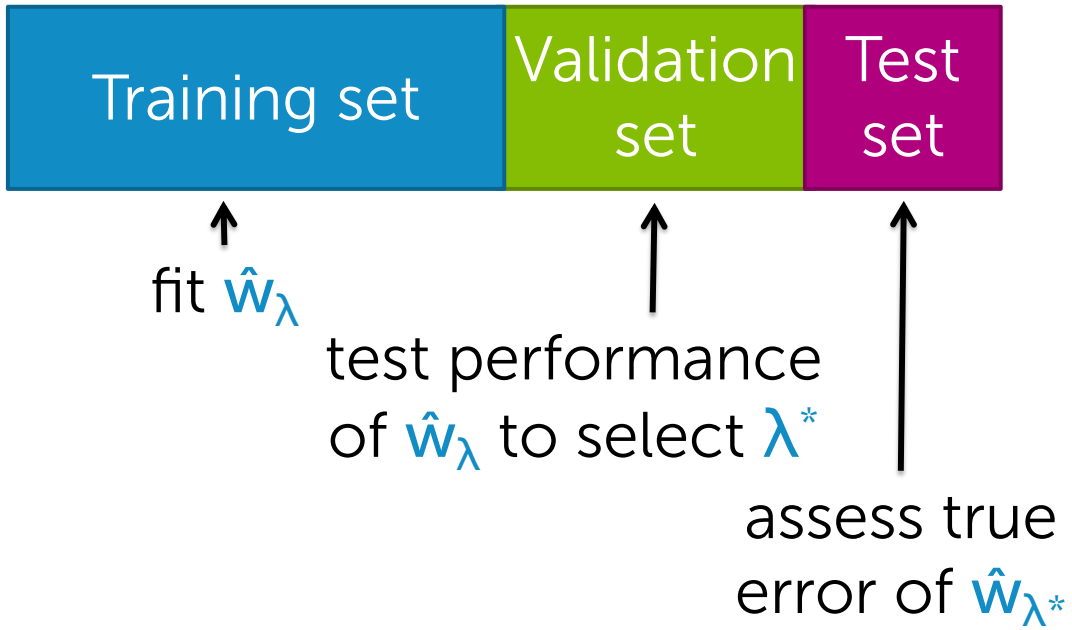
- λ^* was selected to minimize test error (i.e., λ^* was fit on test data)
- If test data is not representative of the whole world, then $\hat{w}_{\lambda^*}$ will typically perform worse than test error indicates

Practical implementation

Solution: Create two "test" sets!

1. Select λ^* such that $\hat{w}_{\lambda^*}$ minimizes error on validation set
2. Approximate true error of $\hat{w}_{\lambda^*}$ using test set

Practical implementation



Feature normalization

Normalizing features

Scale training **columns** (not rows!) as:

$$\underline{h_j(x_k)} = \frac{h_j(x_k)}{\sqrt{\sum_{i=1}^{N} h_j(x_i)^2}}$$

Normalizer: Z_j

Apply same training scale factors to test data:

$$\underline{h_j(x_k)} = \frac{h_j(x_k)}{\sqrt{\sum_{i=1}^{N} h_j(x_i)^2}}$$

apply to test point

Normalizer: Z_j

summing over training points

Summary for ridge regression

What you can do now…

- Describe what happens to magnitude of estimated coefficients when model is overfit
- Motivate form of ridge regression cost function
- Describe what happens to estimated coefficients of ridge regression as tuning parameter λ is varied
- Interpret coefficient path plot
- Use a validation set to select the ridge regression tuning parameter λ
- Handle intercept and scale of features with care

Annotated ridge Slides

Ridge Regression:

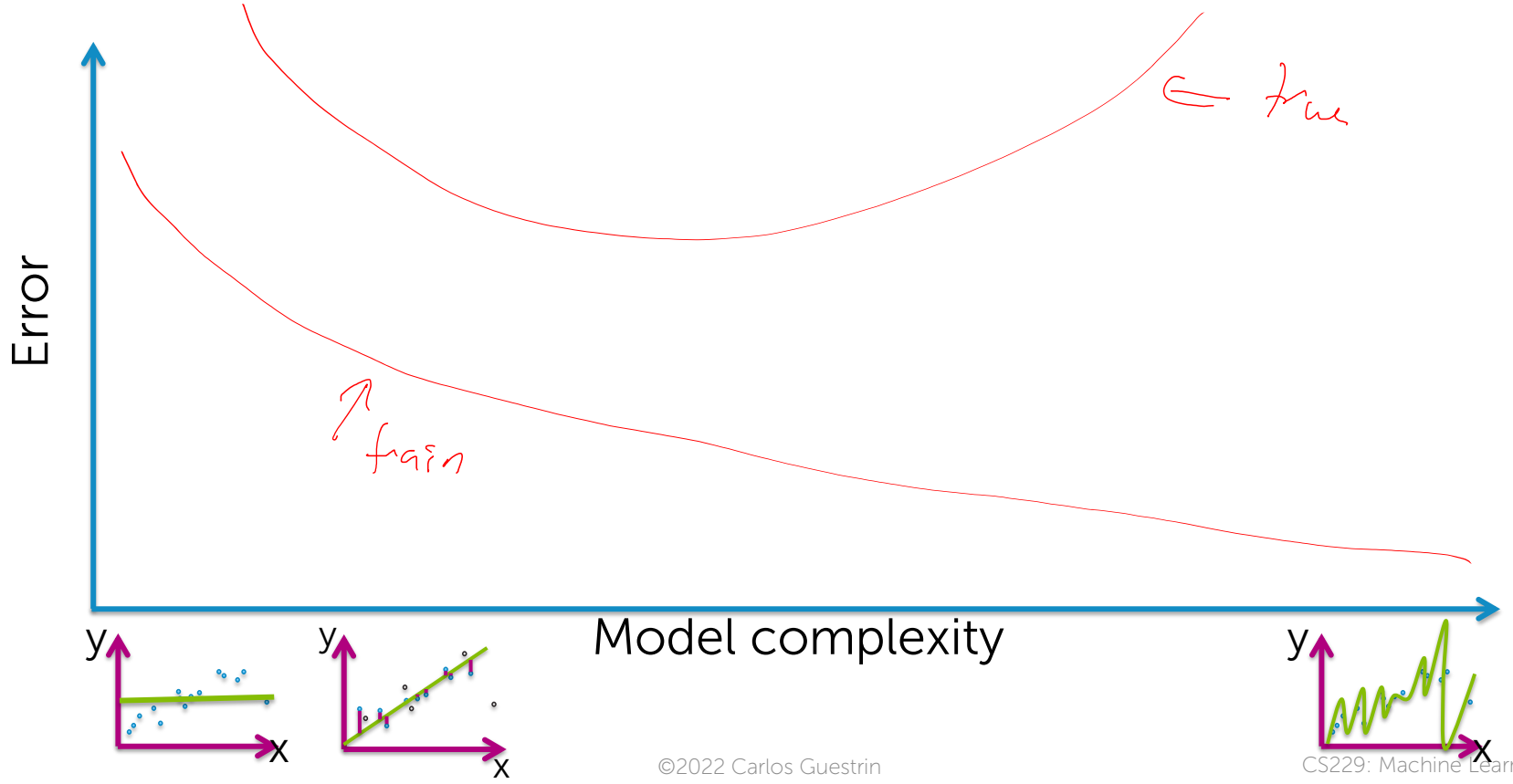
Regulating overfitting when using many features

CS229: Machine Learning

Carlos Guestrin
Stanford University

Slides include content developed by and co-developed with Emily Fox

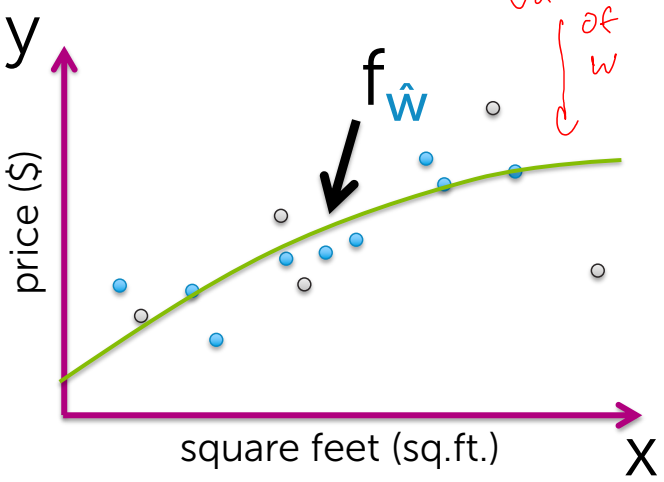
Training, true vs. model complexity

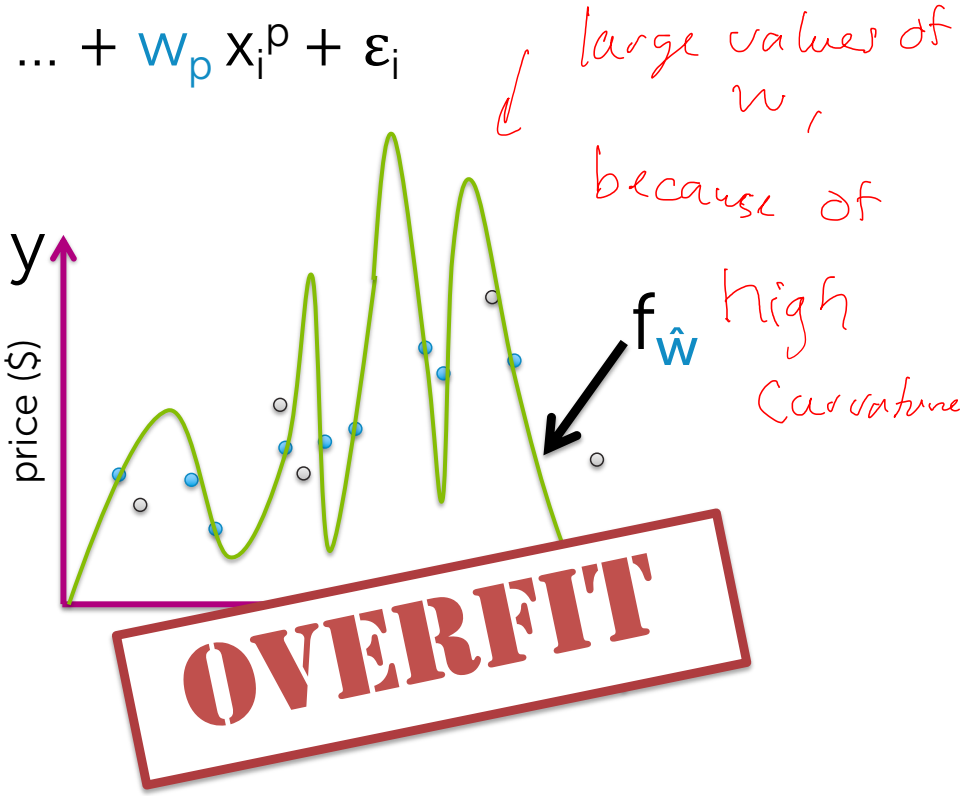


Overfitting of polynomial regression

Flexibility of high-order polynomials

$$y_i = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p + \varepsilon_i$$

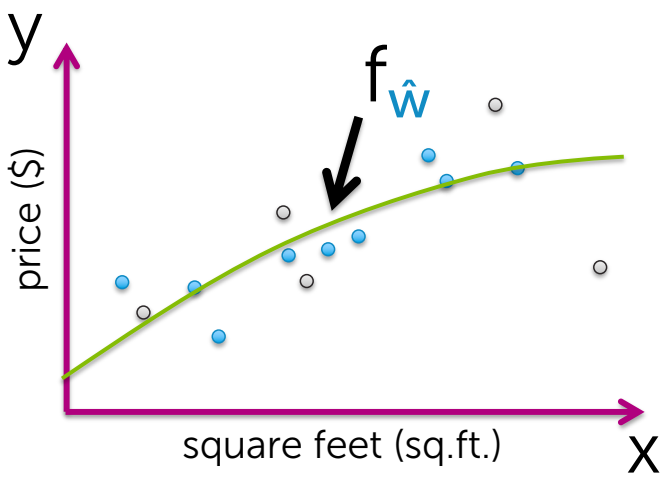




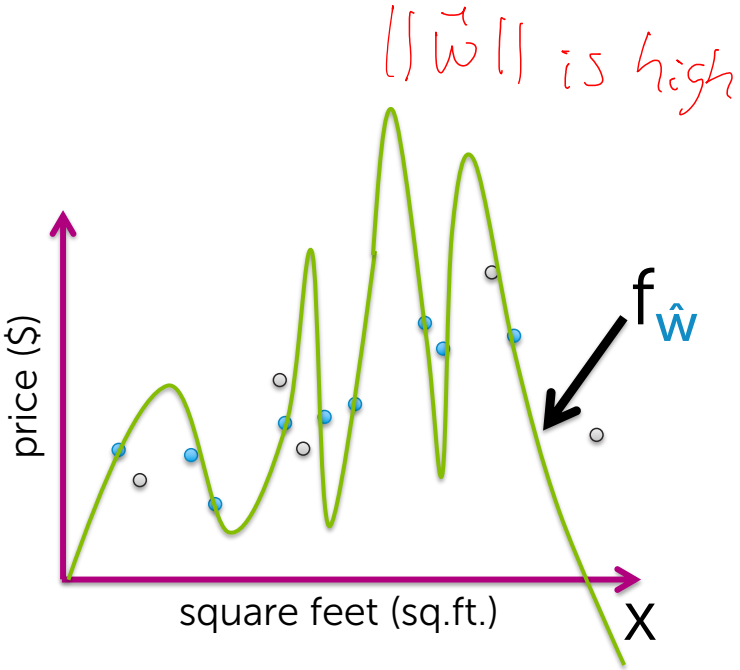
Symptom of overfitting

Often, overfitting associated with very large estimated parameters $\hat{w}$

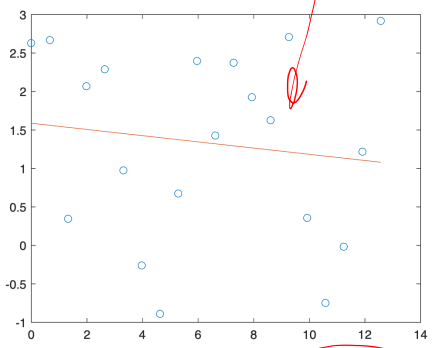
$\|\hat{w}\|$ is low



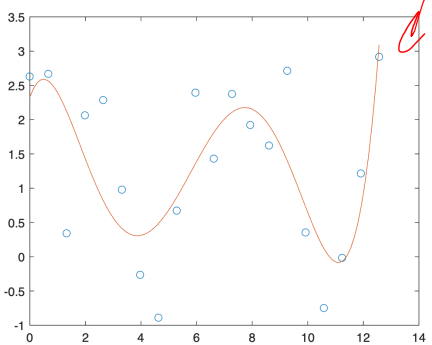
$\|\hat{w}\|$ is high



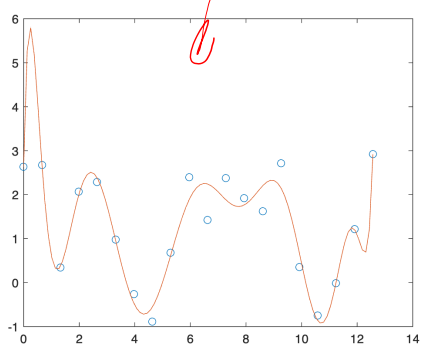
Polynomial fit example



w0 = -0.0403609 w1 = 1.58766



w0 = 0.00142109 w1 = -0.0412048 w2 = 0.402433
w3 = -1.45804 w4 = 1.16305 w5 = 2.32569



w0 = -3.33355e-09 w1 = 3.24407e-07 w2 = -1.3957e-05 w3 = 0.000351859 w4 = -0.00580734
w5 = 0.0664276 w6 = -0.543967 w7 = 3.24647 w8 = -14.1922 w9 = 44.8987 w10 = -98.886
w11 = 139.912 w12 = -109.084 w13 = 32.5699 w14 = 2.62986

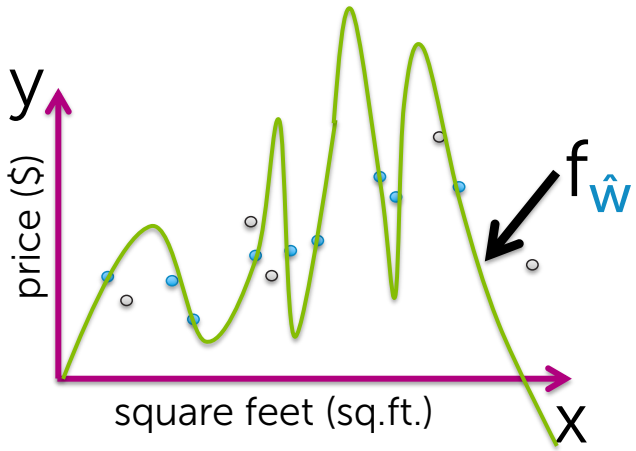
How does # of observations influence overfitting?

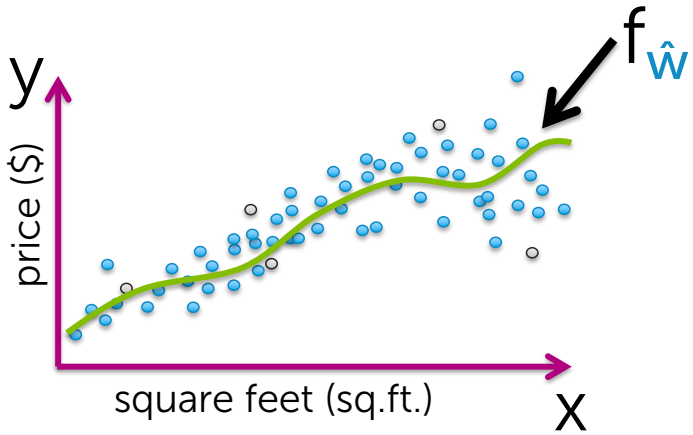
Few observations (N small)

→ rapidly overfit as model complexity increases

Many observations (N very large)

→ harder to overfit





Overfitting of linear regression models more generically

Overfitting with many features

Not unique to polynomial regression,
but also if **lots of inputs** (d large)

- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- ...

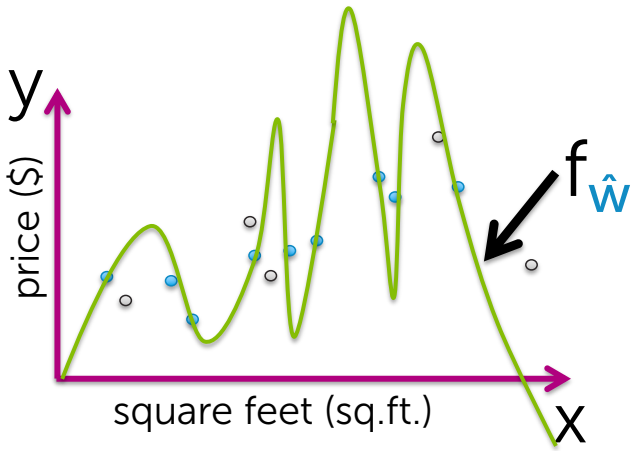
Or, generically,
lots of features (D large)

$$y = \sum_{j=0}^{D} w_j h_j(x) + \varepsilon$$

How does # of inputs influence overfitting?

1 input (e.g., sq.ft.):

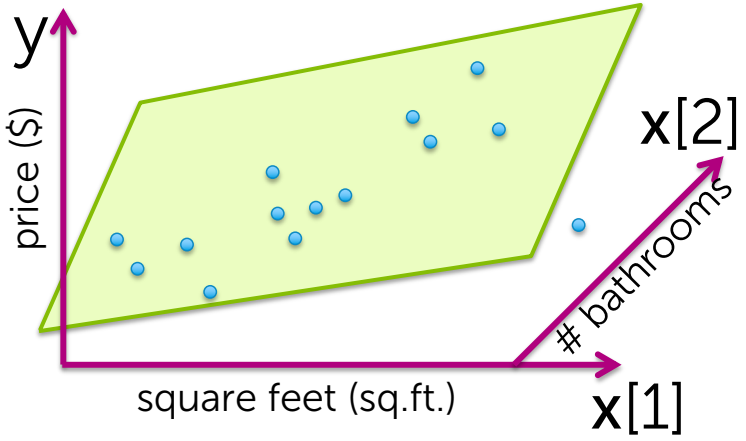
Data must include representative examples of all possible (sq.ft., $) pairs to avoid overfitting



How does # of inputs influence overfitting?

d inputs (e.g., sq.ft., #bath, #bed, lot size, year,…):

Data must include examples of all possible (sq.ft., #bath, #bed, lot size, year,…., $) combos to avoid overfitting



Regularization:
Adding term to cost-of-fit to prefer small coefficients

Desired total cost format

Want to balance:

i. How well function fits data

ii. Magnitude of coefficients

Total cost =

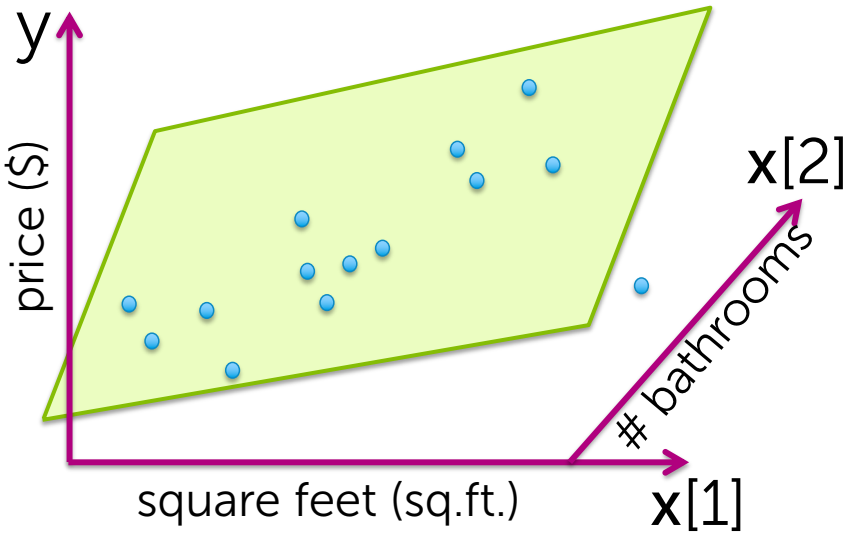
measure of fit + measure of magnitude of coefficients

e.g., RSS

penalty for large parameter values

force me to find simpler models

Measure of fit to training data



$$\text{RSS}(\mathbf{w}) \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T \mathbf{w})^2$$

Measure of magnitude of regression coefficient

What summary # is indicative of size of regression coefficients?

- Sum? $w_0 + w_1 + w_2 + \cdots$ typically bad (negative w) ideal ?
- Sum of absolute value? $\|w\|_1 = |w_0| + |w_1| + |w_2| + \cdots$ (Lasso)
- Sum of squares (L_2 norm) $\|w\|_2^2 = w_0^2 + w_1^2 + w_2^2 + \cdots$ (ridge regression)

Consider specific total cost

Total cost =

measure of fit + measure of magnitude of coefficients

$RSS(w) + \lambda \|w\|_2^2$

λ: magic tradeoff parameter

Ridge Regression (aka L_2 regularization)

What if $\hat{w}$ selected to minimize

$$\text{RSS}(w) + \lambda \|w\|_2^2$$

If $\lambda=0$: $\hat{w}_{ridge} = \hat{w}_{RSS}$

If $\lambda=\infty$: $\hat{w}_{ridge} = 0$

If λ in between: $\|\hat{w}_{ridge}\|_2^2 < \|w_{RSS}\|_2^2$

Bias-variance tradeoff

Large λ:

high bias, low variance

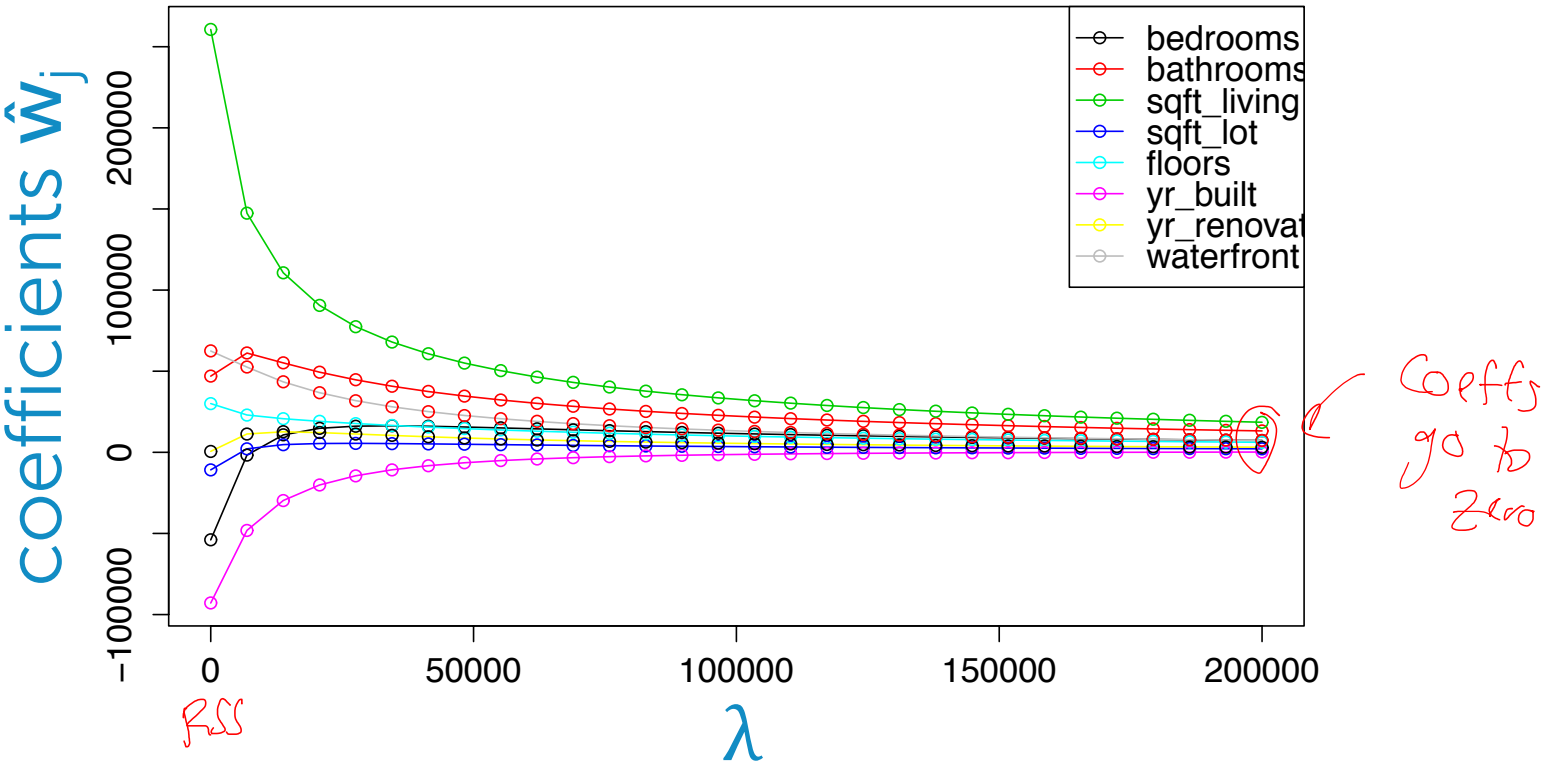
(e.g., $\hat{w} = 0$ for $\lambda = \infty$)

Small λ:

low bias, high variance

(e.g., standard least squares (RSS) fit of high-order polynomial for $\lambda = 0$)

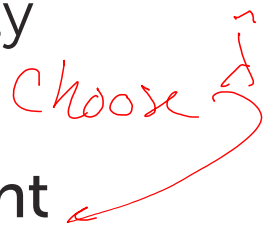
Coefficient path



How to choose λ

The regression/ML workflow

1. **Model selection**
   Need to choose tuning parameters λ controlling model complexity



2. **Model assessment**
   Having selected a model, assess generalization error

Hypothetical implementation 1

1. **Model selection**

For each considered λ :

i. Estimate parameters $\hat{w}_\lambda$ on **training data**

ii. Assess performance of $\hat{w}_\lambda$ on **training data**

iii. Choose λ^* to be λ with lowest train error

2. **Model assessment**

Compute test error of $\hat{w}_{\lambda^*}$ (fitted model for selected λ^*) to approx. true error

Hypothetical implementation 1

Issue: Both λ and $\hat{w}$ selected on training data then $\lambda^* = 0$

- λ^* was selected to minimize training error (i.e., λ^* was fit on training data)
- Most complex model will have lowest training error

Hypothetical implementation 2

1. **Model selection**

For each considered λ :

i. Estimate parameters $\hat{w}_\lambda$ on **training data**

ii. Assess performance of $\hat{w}_\lambda$ on **test data**

iii. Choose λ^* to be λ with lowest test error

2. **Model assessment**

Compute test error of $\hat{w}_{\lambda^*}$ (fitted model for selected λ^*) to approx. true error

Hypothetical implementation 2

Issue: Just like fitting $\hat{w}$ and assessing its performance both on training data

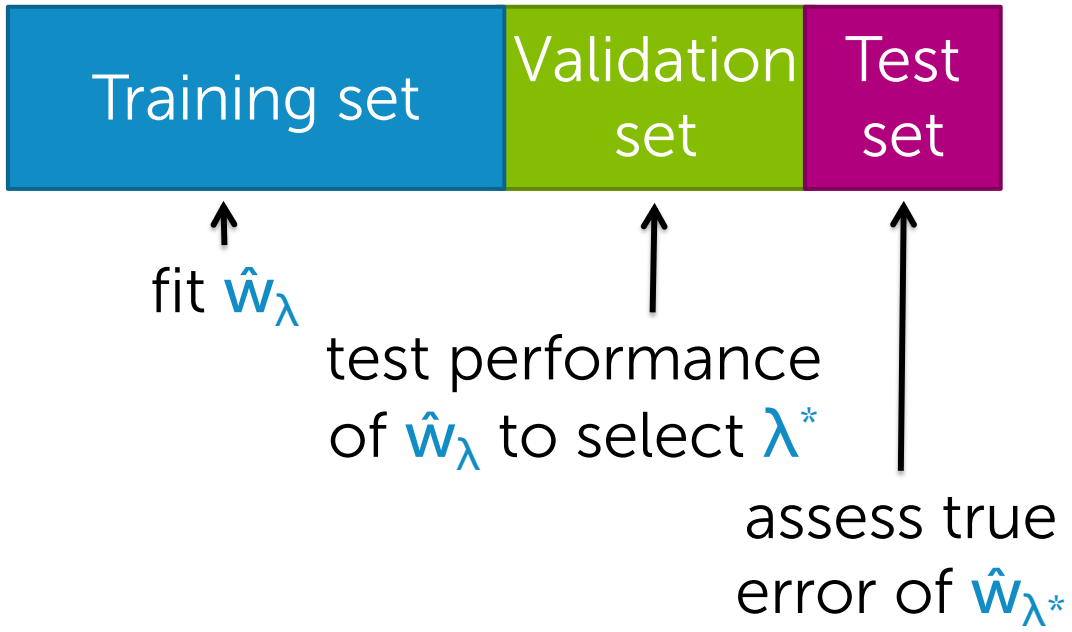
- λ^* was selected to minimize test error (i.e., λ^* was fit on test data)
- If test data is not representative of the whole world, then $\hat{w}_{\lambda^*}$ will typically perform worse than test error indicates

Practical implementation

Solution: Create two "test" sets!

1. Select λ^* such that $\hat{w}_{\lambda^*}$ minimizes error on validation set
2. Approximate true error of $\hat{w}_{\lambda^*}$ using test set

Practical implementation



Feature normalization

Normalizing features

Scale training **columns** (not rows!) as:

$$\underline{h_j(x_k)} = \frac{h_j(x_k)}{\sqrt{\sum_{i=1}^{N} h_j(x_i)^2}}$$

Normalizer: Z_j

Apply same training scale factors to test data:

$$\underline{h_j(x_k)} = \frac{h_j(x_k)}{\sqrt{\sum_{i=1}^{N} h_j(x_i)^2}}$$

apply to test point

Normalizer: Z_j

summing over training points

Summary for ridge regression

What you can do now...

- Describe what happens to magnitude of estimated coefficients when model is overfit
- Motivate form of ridge regression cost function
- Describe what happens to estimated coefficients of ridge regression as tuning parameter λ is varied
- Interpret coefficient path plot
- Use a validation set to select the ridge regression tuning parameter λ
- Handle intercept and scale of features with care